ISAP – Authenticated Encryption Inherently Secure Against Passive Side-Channel Attacks

Christoph Dobraunig, Maria Eichlseder, Stefan Mangard,
Florian Mendel, and Thomas Unterluggauer

Graz University of Technology, Austria
{firstname.lastname}@iaik.tugraz.at

Abstract. Side-channel attacks and in particular differential power analysis (DPA) attacks pose a serious threat to cryptographic implementations. One approach to counteract such attacks are cryptographic schemes based on fresh re-keying. In settings of pre-shared secret keys, such schemes render DPA infeasible by deriving session keys and by ensuring that the attacker cannot collect side-channel leakage on the session key during cryptographic operations with different inputs. While these schemes can be applied to secure standard communication settings, current re-keying approaches are unable to provide protection in settings where the same input needs to be processed multiple times.
In this work, we therefore adapt the re-keying approach to present the first symmetric authenticated encryption scheme that is inherently secure against DPA attacks and that does not have such a usage restriction. This means that our scheme fully complies with the requirements given in the CAESAR call and hence, can be used like other standard authenticated encryption schemes without loss of side-channel protection. Its resistance against side-channel analysis is highly relevant for several applications in practice, like bulk storage settings in general and the protection of FPGA bitfiles and firmware images in particular.

Keywords: authenticated encryption · fresh re-keying · passive side-channel attacks · sponge construction · permutation-based construction

1 Introduction

Motivation. Passive side-channel attacks pose a serious threat to the security of cryptographic implementations. These attacks allow to learn information about the secret key that is processed inside a device by observing physical properties, like the timing, the power consumption [20], the electromagnetic (EM) field [34],... or even several of them at the same time. While passive side-channel attacks have mainly been a threat to ATM and pay TV cards at the time of their publication, these attacks are now relevant to a wide range of devices of the Internet of Things (IoT).

Side-channel attacks pose a threat whenever a device performs cryptographic operations with a key that is not known to the holder of a device. This is for

example the case, when a sensor device is installed in a non-protected area to communicate data to some backend, when a manufacturer performs an encrypted firmware update on devices in the field, when a device working on encrypted data is lost, or when a device is rented by one party to another.

Among others, prominent attacks in such settings have been performed on the encryption of FPGA bitfiles [26,27]. In fact, the keys for bitfile encryption of several generations of FPGAs have been revealed using side-channel attacks. While performing side-channel attacks on FPGAs typically requires a standard oscilloscope, there are now also open source projects offering low-cost measurement setups to attack software implementations [28]. Using these setups, unprotected software implementations of cryptographic algorithms can typically be broken by observing less than 100 encryptions or decryptions using the same key [21]. Given the low effort of the attacks, there is a great need for countermeasures.

State of the Art. In order to protect cryptographic keys against side-channel attacks, a lot of research has been conducted during the last two decades. Today, there essentially exist two approaches to counteract such attacks. The first approach works by hardening the implementation of cryptographic algorithms with techniques like hiding [21] and masking [33]. The goal of masking is to randomize the power consumption of a device by randomizing the intermediate results of the executed cryptographic algorithm. However, the cost of masking increases quadratically with the protection order [19].

The second approach to counteract side-channel attacks is to change cryptographic protocols in such a way that certain types of side-channel attacks cannot be performed at all on the underlying cryptographic primitive. Most protocol designs aim at inherently preventing differential power analysis (DPA) attacks, which is the strongest class of passive side-channel attacks. DPA attacks accumulate information about a cryptographic key by observing multiple encryptions/decryptions of different inputs. The fact that different inputs are used allows to extract keys very efficiently via statistical techniques like Bayesian distinguishers [12] or correlation [11]. In case DPA attacks are prevented by the design of the protocol, the basic approach thus is to limit the exploitable data complexity of the underlying cryptographic primitive for a each key by a certain number q (q-limiting [35]). The corner case are 1-limiting constructions which are *inherently secure against DPA attacks* as they allow attackers to just observe one input per secret key. Hence, attackers are restricted to techniques like simple power analysis (SPA) which eventually leads to significantly lower overheads for the implementation of the cryptographic primitive.

Examples of the approach of inherently preventing DPA attacks are fresh re-keying [22, 23] and leakage-resilient cryptography, which brought forth encryption schemes [17,31] and message authentication codes (MACs) [29]. While the schemes as such are quite different, the security of all the published schemes with inherent protection against DPA attacks relies on two basic properties. First, the schemes require a side-channel secure initialization with a fresh session key on every invocation. Second, the schemes require that information an

attacker can learn by collecting side-channel information about the session key is bounded [16]. These two basic properties do not just guarantee side-channel security, but also result in designs that turn out to be quite efficient for processing bulk data, since—besides the side-channel secure initialization—there are no DPA requirements on the implementation of the cryptographic primitive.

In this work, we present a novel symmetric authenticated encryption scheme that also relies on these two basic properties, but which provides significant improvements with respect to both properties. First, our authenticated encryption scheme can be applied to settings where it is highly beneficial, or even required, to allow multiple decryptions and verifications of the same ciphertext. Current schemes have not been designed to be used in such settings. Second, we show that the parameters of permutation-based cryptographic primitives provide a flexible tool to cope with the maximum tolerable leakage of cryptographic schemes on an algorithmic level.

Our Contribution. We now detail the two main contributions of the article. Schemes with inherent protection against DPA require a side-channel secure initialization in order to obtain a fresh session key for every cryptographic operation. Such session key k_0 is typically derived from a pre-shared master key K using a nonce n by means of a re-keying function $g : (K, n) \mapsto k_0$ that is carefully designed to prevent both DPA and SPA attacks. The purpose of this secure initialization is to ensure that cryptographic operations for different data inputs are always done using different keys. Hence, whenever a party encrypts or authenticates data, a new nonce is generated to derive a new session key.

While this approach successfully prevents DPA on the party performing the encryption or authentication (sender of a message), the situation is more challenging for the party performing decryptions or verifications (receiver of a message). The reason for this is that the decrypting party has no control of the nonce n. Therefore, an attacker might send arbitrary messages to the decryption device using the same nonce n for all sent messages. This behavior results in different messages being decrypted using the same session key k_0. As a result, decryption is vulnerable to DPA, and more concretely, it is the multiple decryption with the same session key k_0 that causes this DPA vulnerability. In order to prevent this kind of DPA attacks, the receiver either needs to be protected by other means [23], or all communicating parties are required to contribute to the nonce that is used to derive the session key from a pre-shared master key [22].

In our first contribution, we overcome this problematic situation and present a symmetric authenticated encryption scheme that does not have any special requirements on the initialization and the nonces. In fact, with respect to both usage and requirements, it is a standard nonce-based symmetric authenticated encryption scheme that fulfills all functional requirements of the CAESAR call [36] and at the same time provides inherent protection against DPA attacks for all involved parties, i.e., also the decrypting party. This is achieved by making the initialization of the authenticated encryption scheme depend on the processed message itself. This implicitly prevents DPA attacks and enables several new

use cases. In particular, the scheme remains secure in settings that allow multiple decryptions and verifications of the same ciphertext. Such settings cannot be realized with existing schemes, as they would require fresh nonces for every encryption and decryption.

Our second contribution is that we show how permutation-based designs can be used in order to scale implementations for different leakage bounds. Essentially, we model the side-channel leakage as part of the public output of the permutation. This allows us to adjust the maximum tolerable leakage by varying the permutation parameters. Using this flexible tool as a basis, we propose an efficient sponge-based variant of our authenticated encryption scheme and two novel permutation-based re-keying functions inherently secure against DPA attacks. We instantiate the sponge-based authenticated encryption scheme Isap using Keccak[400] and present the results of its hardware implementation. This instance maintains 128-bit security in the presence of up to 16 bits leakage per permutation call, can be used in settings of multiple decryptions and verifications, and yet has approximately the same runtime and area requirements as state-of-the-art schemes. Put into numbers, the hardware implementation using an UMC 130 nm technology consumes 14 kGE, takes 22.35μs for secure initialization, and performs authenticated encryption in roughly 0.15μs per 128-bit block.

All these properties make Isap suitable for a set of highly relevant settings in practice. One prominent example is the decryption and verification of firmware images or FPGA bitfiles, which requires that it is possible to do multiple decryptions and verifications with the same session key by different parties. There is one party that encrypts and authenticates the image or bitfile once and there are many devices that decrypt and verify it. Another example is the bulk storage of data. In such scenario, the goal is to encrypt and authenticate once and to allow multiple decryptions and verifications without the need to re-encrypt the data upon every read operation. These scenarios again highlight the main benefit of Isap over existing schemes: Isap remains secure in all these settings allowing for multiple decryption and verification and simultaneously has practical implementation cost.

Outline. We introduce our new symmetric authenticated encryption scheme that is also secure in the setting of multiple decryptions and verifications in Section 2. Section 3 shows how permutation-based designs can be used to achieve scalable security against side-channel attacks and gives an permutation-based instance of our authenticated encryption scheme. Section 4 concludes this work. In addition, we give details about the Keccak[400]-based instance of our authenticated encryption scheme and results of its hardware implementation in the appendix.

2 Authenticated Encryption Mode

State-of-the-art schemes based on re-keying lack security against DPA in scenarios that require multiple decryption of the same input (with the same session key). Therefore, this section presents the first nonce-based authenticated encryption mode that is designed to be secure against passive side-channel attacks in such scenarios. To achieve this inherent security against DPA attacks, our authenticated encryption mode incorporates the re-keying approach [22, 23] in an efficient encrypt-then-MAC scheme. The resulting nonce-based two-pass authenticated encryption mode fulfills the functional requirements of the CAESAR call [36] without any additional considerations. Especially the fact that the nonce has to be unique per encryption is covered by the requirements of CAESAR. Hence, our mode can be used in all applications that use an authenticated encryption algorithm in a nonce-respecting way. In this section, we first specify the authenticated encryption mode and afterwards discuss its resistance against side-channel attacks, especially its inherent DPA security.

2.1 Specification

The inputs for the authenticated encryption $\mathcal{E}$ are the secret key $K_1 \| K_2$, a unique public message number (nonce) n, associated data a, and plaintext p. The output of the authenticated encryption $\mathcal{E}$ is a ciphertext c and a tag t. Note that the nonce ideally is not input by the user but generated directly on the device implementing the authenticated encryption $\mathcal{E}$.

$$\mathcal{E}(K_1 \| K_2, n, a, p) = (c, t)$$

The authenticated decryption $\mathcal{D}$ (decryption plus verification) takes as input the key $K_1 \| K_2$, nonce n, associated data a, ciphertext c, and tag t, and outputs the plaintext p, or $\perp$ if the verification fails:

$$\mathcal{D}(K_1 \| K_2, n, a, c, t) \in \{p, \perp\}$$

As shown in Algorithm 1, the basis of our authenticated encryption mode is the generic composition encrypt-then-MAC. Bellare and Namprempre [3] showed that this generic composition is capable of achieving the strongest security notions. Here, *ENC* and *DEC* denote a nonce-based SPA-secure encryption and decryption scheme, *MAC* denotes an SPA-secure message authentication code, H denotes a cryptographic hash function with distinct inputs for nonce, associated data and ciphertext, and g_1 and g_2 are secure re-keying functions (secure against SPA and DPA). Note that g_1 and g_2 are not required to be distinct.

2.2 Side-channel Resistance

In this section, we discuss the security of our authenticated encryption mode against passive side channel attacks, and in particular DPA. Hereby, our approach essentially uses the same assumptions and requirements as other re-keying schemes. Namely, we assume g_1, g_2 to be (DPA and SPA) secure re-keying

Algorithm 1: Authenticated encryption and decryption procedures.

Auth. Encryption $\mathcal{E}(K_1 \| K_2, n, a, p)$	Auth. Decryption $\mathcal{D}(K_1 \| K_2, n, a, c, t)$
Input: keys $K_1 \in \{0,1\}^m$, $K_2 \in \{0,1\}^m$, public message number $n \in \{0,1\}^m$, associated data $a \in \{0,1\}^*$, plaintext $p \in \{0,1\}^*$	**Input:** keys $K_1 \in \{0,1\}^m$, $K_2 \in \{0,1\}^m$, public message number $n \in \{0,1\}^m$, tag $t \in \{0,1\}^m$, associated data $a \in \{0,1\}^*$, ciphertext $c \in \{0,1\}^*$
Output: ciphertext $c \in \{0,1\}^*$, tag $t \in \{0,1\}^m$	**Output:** plaintext $p \in \{0,1\}^*$, or $\perp$
Encryption $k_1 = g_1(n, K_1)$ $c = ENC_{n,k_1}(p)$ **Authentication** $y = H(n, a, c)$ $k_2 = g_2(y, K_2)$ $t = MAC_{k_2}(y)$ **return** c, t	**Verification** $y = H(n, a, c)$ $k_2 = g_2(y, K_2)$ $t' = MAC_{k_2}(y)$ **if** $t \neq t'$ **return** $\perp$ **Decryption** $k_1 = g_1(n, K_1)$ $p = DEC_{n,k_1}(c)$ **return** p

functions and assume the implementations of *ENC*, *DEC*, and *MAC* to have bounded SPA leakage such as in [15–17, 29, 31, 37], i.e., *ENC*, *DEC*, and *MAC* must be able to process arbitrarily long messages within a given leakage bound to prohibit SPA attacks. However, there are no requirements on the implementation of the hash function H, since it processes only publicly known data.

We now show that our mode is secure against DPA by discussing the security of the en-/decryption part and the authentication part separately.

Encryption/Decryption. The encryption and decryption part is an instance of fresh-rekeying such as in [22, 23]. Such schemes for fresh re-keying combine an SPA-secure encryption scheme *ENC* with a (DPA and SPA) secure re-keying function $g_1 : (K, n) \mapsto k_1$. As the nonce n that is used to derive the session key k_1 must not be repeated, fresh session keys are guaranteed and DPA on the encryption scheme *ENC* is inherently prevented.

However, for decryption, there is the threat that an adversary could exploit multiple decryptions (using different data) with the same session key k_1 to induce a DPA setting within the decryption *DEC*, since multiple calls of *DEC* with the same nonce n are allowed. To prevent such a DPA scenario in our mode, authentication is performed prior to decryption. The authentication part aborts the process if tag verification fails, which ensures that the same session key k_1 is never used to decrypt distinct ciphertexts. Therefore, the authentication part precludes DPA attacks on the decryption part.

Authentication/Verification. The authentication and verification part can be seen as a tweaked instance of the re-keying concept. Hereby, the session key k_2 is derived from the hash value y that is computed from the nonce n,

associated data a, and ciphertext c using a cryptographic hash function H. For every authenticated encryption $\mathcal{E}$, a new unique nonce n has to be used. This results in a new session key k_2 to be used to generate the tag t, since unique nonces n result in unique hash values y in the absence of collisions. Hence, DPA attacks on the MAC are inherently prevented during the generation of the tag t as the same session key k_2 is never used to authenticate distinct ciphertexts.

Considering the decryption and verification $\mathcal{D}$, we cannot rely on the uniqueness of the nonce n anymore. An attacker can usually modify the associated data a, or ciphertext c, to provoke multiple decryptions and verifications $\mathcal{D}$ with different data under the same nonce n. However, such a DPA scenario on the MAC session key k_2 is prevented in our mode as k_2 is bound to the data it processes. Namely, as y depends on n, a and c, the MAC session key $k_2 = g_2(y, K_2)$ changes whenever the data changes. This ensures that modifications to the input data cause the tag verification to fail and the process to abort before the actual decryption DEC of the ciphertext starts.

Moreover, no adversary can predictably influence y due to the use of a cryptographic hash function H. This guarantees that the key k_2 is unique for every new triple of nonce n, associated data a, and ciphertext c, as long as there is neither a collision in the hash y nor in the re-keying function g_2. Thus DPA on the MAC session key k_2 during decryption and verification $\mathcal{D}$ is inherently prevented.

Note however that collisions in the re-keying function g_2 or the hash function H result in the same session key k_2 being used in MAC computations of different data, thus allowing for a DPA. Yet, collisions in g_2 depend on the secret key K_2 and therefore inputs causing collisions in g_2 cannot be calculated off-line. Moreover, choosing g_2 to be a permutation for a fixed key K_2 renders collisions in g_2 infeasible. In contrast, collisions in the hash value y are directly observable and can be calculated off-line. The complexity of calculating collisions off-line is determined by the size of the hash. The generic complexity of finding a collision for an m-bit hash function is $2^{m/2}$. Hence, the size of the hash needs to be chosen depending on the potential threat of such an event. This will be discussed in more detail in the next section.

3 ISAP

Permutation-based cryptographic designs have become quite popular in the last years. For example, three out of five finalists in the SHA-3 hash competition were based on permutations, with the most prominent example being the sponge construction [6] of SHA-3 winner Keccak [7]. The versatile sponge construction was consequently adopted in the design of various cryptographic primitives, e.g., in authenticated encryption (AE) designs for the ongoing CAESAR competition.

Besides their flexibility, permutation-based constructions also offer a convenient way to deal with bounded SPA leakage. As we will show in this section, especially the sponge construction allows simple and elegant arguments on the side-channel security assuming a bounded leakage of the underlying permutation.

In this section, we therefore introduce ISAP – an authenticated encryption scheme inherently secure against passive side-channel attacks that solely relies on permutation-based primitives. Based on the ideas in Section 2, ISAP consists of both a sponge-based encryption scheme ISAPENC and a sponge-based authentication part ISAPMAC. ISAPENC was designed as a streaming mode since previous work [17,31] already suggests the high suitability of streaming modes to obtain encryption schemes secure against side-channel attacks. On the other hand, ISAPMAC combines a re-keying function and a sponge-based MAC in a novel way to obtain a MAC inherently secure against DPA with only one pass over the input data. For the secure re-keying function, either a generic permutation-based construction, ISAPRK1, or a sponge construction, ISAPRK2, can be used. ISAPRK1 is a design inherently secure against DPA, whereas ISAPRK2 is a more efficient design based on a stronger side-channel assumption.

Throughout this section, the side-channel discussion of the four ISAP primitives assumes SPA secure implementations of the single components and focuses on DPA only. A discussion on the SPA security of the four primitives and the underlying permutations in general follows at the end of this section.

3.1 Authenticated Encryption Mode

The sponge-based instances of the encryption part ISAPENC and the authentication part ISAPMAC are now presented consecutively.

Encryption/Decryption. The sponge mode to encrypt plaintexts, ISAPENC, is shown in Fig. 1. It is an adaptation of the streaming mode in [6], which is proven cryptographically secure in [1]. In contrast to the "standard" sponge-based streaming mode in [6], ISAPENC uses a different session key k_1 for each new nonce n. This session key k_1 is provided via the secure re-keying function g_1.

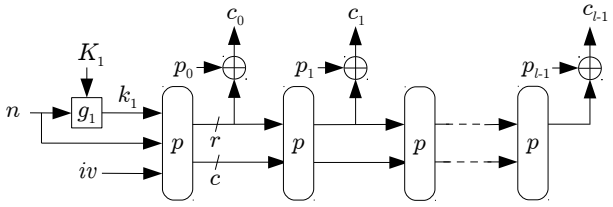


Fig. 1: Encryption part: ISAPENC.

Whenever a cryptographic primitive is frequently re-keyed, care has to be taken to preclude generic time-memory trade-off (TMTO) attacks to recover the secret master key K_1 [13]. To avoid such attacks, ISAPENC uses both the session key k_1 and the nonce n as inputs to the first permutation call p. Hence,

the design principle is similar to the fresh re-keying schemes recently presented in [14].

The initialization with a fresh nonce n and a new session key k_1 for each encryption ensures that the key stream is unpredictable and unique for different encryptions. Multiple decryption of different ciphertexts with the same nonce n and session key k_1 is inherently prevented by the authentication part as discussed in Section 2. DPA on the master key K_1 is prevented by the use of the (DPA and SPA) secure re-keying function g_1 to initialize the streaming mode in IsapEnc.

Authentication/Verification. The authentication part of the authenticated encryption mode in Algorithm 1 consists of the following three steps:

1. Hash the data to get y,
2. Use y to derive the data-dependent MAC session key k_2, and
3. Compute the MAC with k_2 to authenticate the data.

Following this description, two cryptographic primitives, a hash function and a MAC, are required. However, a suffix-MAC allows to virtually combine the hash function and the MAC in one primitive. The result is IsapMac in Fig. 2, a sponge-based suffix-MAC that is inherently secure against DPA.

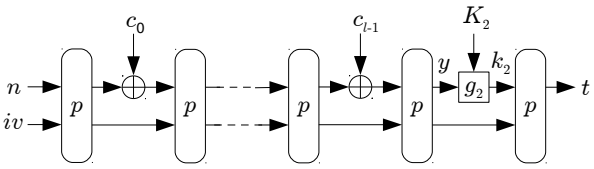


Fig. 2: Authentication part: IsapMac (not showing authenticated data).

Bertoni et al. [6] showed that one can always turn a sponge into a MAC by either putting the key before (prefix-MAC), or after the message (suffix-MAC), as this always gives a pseudo-random function as long as the sponge itself behaves like a random oracle. Compared to a "standard" sponge-based suffix-MAC, IsapMac uses a secure re-keying function g_2 to absorb the secret key K_2. Note however, that whenever a suffix-MAC is used, care has to be taken with the choice of the parameters and the padding rule to preclude some generic attacks [32].

IsapMac prevents DPA on the tag computation in two ways. First, and as shown in Fig. 2, the MAC session key k_2 is derived from the hash value y and the MAC master key K_2 via a secure re-keying function g_2, thus prohibiting DPA on K_2. Second, the design inherently prevents DPA on the MAC session key k_2 by binding it to the data being processed, thus leading to different MAC session keys k_2 for different data.

As already mentioned in Section 2.2, a collision in the hash value y allows for two side-channel measurements of the MAC using different data but the

same MAC session key k_2. This holds true for IsapMac as well. Yet, to perform a successful DPA, usually more than two traces will be needed to recover one fixed session key k_2. Such a setting occurs with hash multi-collisions. The generic complexity for finding a v-collision is $\sqrt[v]{v! \cdot 2^{m(v-1)}}$. Luckily, the complexity is quite high already for small values of v as shown in Table 1 for a 128-bit key. Furthermore, we want to stress that even though a DPA attack exploiting multi-collisions might be able to recover the MAC session key k_2, this does not imply a key recovery attack on the master key K_2 if a non-invertible re-keying function g_2 is used.

Table 1: Complexity for receiving a v-collision for a 128-bit session key k_2.

v	2	3	4	5	$\cdots$	34
complexity	$2^{64.5}$	$2^{86.2}$	$2^{97.1}$	$2^{103.8}$	$\ldots$	2^{128}

3.2 Side-channel Secure Re-keying

Our authenticated encryption scheme requires two re-keying functions $g_1, g_2 : (K, n) \mapsto k$ that are secure against passive side-channel attacks (DPA and SPA). These two functions g_1, g_2 must not necessarily be distinct, but can be the same. We now present two possible options to design such secure re-keying function allowing to reuse the permutation p from our sponge instances IsapEnc and IsapMac. While the first design is inherently secure against DPA attacks, the second design is 2-limiting.

Variant 1. In our first design, we use a variation of the classical GGM construction [18]. The respective re-keying function IsapRk1 is shown in Fig. 3 and works as follows. The state is first initialized with the padded master key K, followed by an application of the permutation p. In each iteration, one bit of the nonce n is processed by either choosing the left or right half of the permutation output, padding it to the permutation size, and again applying the permutation p. Hereby, the padding incorporates information on which half was chosen and on the index of the nonce bit being processed. After all nonce bits have been processed, the session key k is generated from the last permutation output.

The approach to re-keying used in IsapRk1 inherently protects against DPA attacks, since the same secret (i.e., right or left part of the permutation output) is never combined with more than one public input. In this respect, IsapRk1 has a lower data complexity bound than present GGM-based re-keying functions [17, 35] which are 2-limiting when instantiated using common block ciphers [31]. In terms of SPA leakage, a generic treatment of leakage for permutations follows in Section 3.3.

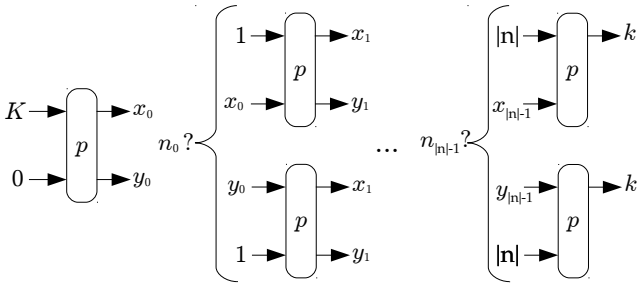


Fig. 3: Re-keying inherently secure against DPA attacks: IsapRk1.

Variant 2. A more efficient re-keying function than IsapRk1 can be obtained from sponges directly, potentially reducing the required state and permutation size. However, the presented re-keying function uses a stronger security assumption than IsapRk1, namely, that DPA is impossible on a 2-limiting primitive, i.e., given the leakages from two different public inputs.

The basic idea is to make DPA infeasible by reducing the input data complexity accordingly. For this purpose, a secret state is constantly updated with small portions of public data by repeating two phases, (1) modifying the secret state according to the public data, and (2) updating the state such that predictions on the future state based on the absorbed public data become infeasible.

Sponges are an ideal choice to implement this basic idea as the rate directly influences the input data complexity for each permutation. Choosing the smallest possible rate ($r = 1$) results in the design IsapRk2 shown in Fig. 4. IsapRk2 first initializes the sponge state by applying the initial permutation p to the padded master key K. Then, IsapRk2 repeatedly injects single nonce bits into the state, each separated by a permutation call. After full absorption of the nonce and a final permutation call, the session key k is output. This working principle is similar to sponge instances of a prefix-MAC. While for general MAC computations the absorption rate can be as big as the state size [8], IsapRk2 uses a small absorption rate in order to limit the data complexity exploitable in a DPA.

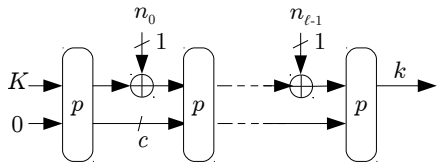


Fig. 4: Sponge construction for re-keying: IsapRk2.

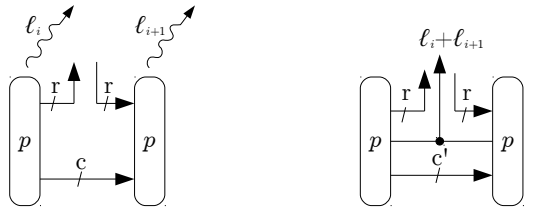
In terms of DPA security, IsapRk2 uses a different assumption than the rest of this paper. For each secret state in IsapRk2, a permutation p will produce

the leakages for two different public inputs. Thus, IsapRk2 is not inherently secure to DPA attacks, but 2-limiting. This results in IsapRk2 being a secure re-keying function under the assumption that the combined leakage resulting from the processing of two different public inputs is bounded such that DPA on the secret state is infeasible. However, note that this construction for a secure re-keying function is again related to the classical GGM construction [18] and can be seen as their sponge equivalent. IsapRk2 is similar to it in the sense that the exploitable data complexity is equal for IsapRk2 and the block-cipher based instantiations of both [17] and the 2PRG primitive used in [35].

3.3 Bounded Leakage in Permutations-based Designs

The permutation-based instances presented before are inherently secure to DPA attacks if used correctly, or, in case of IsapRk2, effectively limit the number of exploitable leakage traces to two. However, it remains to discuss the security of the presented constructions in the presence of SPA leakage. Therefore, we show now that keyed sponges are a convenient tool to argue on the security of permutation-based designs given bounded leakage of the single permutation.

For this purpose, we model the leakage from a permutation p by allowing an adversary to learn a certain amount of the state between subsequent permutation calls as depicted in Fig. 5. Hereby, we use ℓ to denote the amount of information that an attacker can learn about the state from the collected side-channel information in bits. We do not care how and where the leakage is created within p, but let the adversary account the learned information to either the input or the output state of p. Therefore, given two consecutive permutations p with leakages ℓ_i and ℓ_{i+1}, respectively, the maximum an adversary might learn about the state is $\ell_i + \ell_{i+1}$. This means that if each leakage ℓ_i, ℓ_{i+1} is bounded by λ bits and the adversary can optimally combine these two leakages, the adversary will learn at most 2λ bits of the state between the respective two permutation calls.



(a) Leaking permutations (b) Sponge model

Fig. 5: Leakage of information in permutation-based designs.

The basic idea now is to use the sponge parameters to express the constructions capability to cope with the leakage emitted by the permutation. Therefore,

the sponge parameters are adjusted according to the amount of information an adversary learned about the secret state. This means that if the adversary learns 2λ bits of the internal, secret state, the leaked bits can be considered as an increase of the rate, i.e., $r' = r + 2\lambda$, which results in a smaller capacity $c' = c - 2\lambda$ and thus reduced security. However, a reduced security level corresponding to a capacity of $c - 2\lambda$ bits is still guaranteed by the cryptographic properties of the permutation and the associated constrained-input constrained-output (CICO) problem [6]. Therefore, permutation-based designs such as sponges can be considered to have bounded security loss for bounded leakage of the permutation.

Clearly, the challenge in practice is to build an implementation that bounds the leakage of p. However, the advantage of the sponge construction is that besides standard SPA countermeasures, like hiding and masking, the capacity is an additional and very natural security parameter that helps to build a design that withstands side-channel attacks in practice. For example, the leakage assumption in Fig. 5 allows to scale a permutation-based scheme to maintain its desired security level in the presence of λ-bit leakages ℓ_i, ℓ_{i+1} by increasing the capacity by 2λ bits (and either decreasing the rate r accordingly, or increasing the size of the permutation p).

An advantage of this approach is its flexibility. If at some later point the leakage of an existing implementation turns out to be larger, one could simply reduce the rate r to retain the same security level.

In terms of our proposed schemes IsapEnc, IsapMac, and IsapRk2, our assumptions can be straightforwardly applied. With respect to IsapRk1, the modeling works analogously: the 2λ bits learned about the intermediate state account to the known part of the state that without leakage consists of the padding bits. Thus, the size of the permutation p used in IsapRk1 has to be chosen accordingly to obtain a sufficiently large secret part to maintain the desired security level in the presence λ-bit leakage of the permutation.

3.4 Instantiation and Implementation

For the practical use of Isap we propose an instance based on the Keccak[400] permutation. It provides 128-bit security in the presence of up to 16 bits leakage per permutation call. Our parameter choices (permutation size, capacity, rate, number of rounds, etc.) are based on state-of-the-art cryptanalysis results and discussed in detail in Appendix A.

In terms of implementation cost, an UMC-130 nm implementation of Isap with IsapRk2 as the re-keying function consumes merely 14 kGE, takes $22.35\mu s$ for all kind of initialization, and performs authenticated encryption in roughly $0.15\mu s$ per 128-bit block. These hardware results show that Isap extends DPA resistance to settings allowing multiple decryption, resists up to 16 bits leakage per permutation call, and yet yields performance and area figures comparable to state-of-the-art schemes.

These results suggest that Isap is particularly suitable for the encryption and authentication of bulk data, because the scheme does not pose any DPA requirements on the implementation of the cryptographic primitive.

4 Conclusion and Discussion

In this work, we presented a novel nonce-based authenticated encryption mode fulfilling the functional requirements of the CAESAR call. Most notably, this implies that it is not allowed to make any assumptions on the choice of the nonce, besides the fact that the nonce has to be unique per encryption (e.g., it must be possible to implement the nonce as simple counter on encryption side). Our approach ensures that as long as the re-keying function is DPA-secure, both the encrypting and the decrypting entity are also DPA-secure. This is a unique feature of our design compared to existing ones. Consequently, our authenticated encryption mode is highly suitable for scenarios where DPA attacks are a threat and multiple decryptions are useful, or cannot be precluded.

Furthermore, we showed that sponges provide an elegant way to argue the resistance of permutation-based designs to side-channel leakage. This flexibility motivated our permutation-based instances for both the authenticated encryption scheme and the re-keying functions. We give a Keccak[400]-based hardware implementation of Isap. This instance does not only extend security against DPA to multiple decryption with costs comparable to state-of-the-art schemes, but also maintains 128-bit security in the presence of up to 16 bits leakage per permutation call.

Acknowledgments

The authors would like to thank the anonymous reviewers of CHES and CCS for useful comments and discussions.

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 644052 (HECTOR).
Furthermore, this work has been supported in part by the Austrian Research Promotion Agency (FFG) under grant number 845589 and by the Austrian Science Fund (project P26494-N15).

References

1. E. Andreeva, J. Daemen, B. Mennink, and G. Van Assche. Security of keyed sponge constructions using a modular proof approach. In *FSE 2015*, volume 9054 of *LNCS*, pages 364–384. Springer, 2015.
2. S. Belaïd, F. De Santis, J. Heyszl, S. Mangard, M. Medwed, J.-M. Schmidt, F.-X. Standaert, and S. Tillich. Towards fresh re-keying with leakage-resilient PRFs: cipher design principles and analysis. *J. Cryptographic Engineering*, 4(3):157–171, 2014.
3. M. Bellare and C. Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. *J. Cryptology*, 21(4):469–491, 2008.

4. G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. Sponge Functions. ECRYPT Hash Workshop 2007, May 2007.
5. G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. On the indifferentiability of the sponge construction. In *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 181–197. Springer, 2008.
6. G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. Cryptographic sponge functions (Version 0.1). http://sponge.noekeon.org/, 2011.
7. G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. The Keccak SHA-3 submission. http://keccak.noekeon.org/, 2011.
8. G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. Permutation-based encryption, authentication and authenticated encryption. In *DIAC 2012*, pages 159–170. 2012.
9. G. Bertoni, J. Daemen, M. Peeters, G. Van Assche, and R. Van Keer. CAESAR submission: Keyak. http://keyak.noekeon.org/, 2014.
10. A. Bogdanov, C. Dobraunig, M. Eichlseder, M. M. Lauridsen, F. Mendel, M. Schläffer, and E. Tischhauser. Key recovery attacks on recent authenticated ciphers. In *LATINCRYPT 2014*, volume 8895 of *LNCS*, pages 274–287. Springer, 2014.
11. E. Brier, C. Clavier, and F. Olivier. Correlation power analysis with a leakage model. In *CHES 2004*, pages 16–29, 2004.
12. S. Chari, J. R. Rao, and P. Rohatgi. Template attacks. In *CHES 2002*, volume 2523 of *LNCS*, pages 13–28. Springer, 2002.
13. C. Dobraunig, M. Eichlseder, S. Mangard, and F. Mendel. On the security of fresh re-keying to counteract side-channel and fault attacks. In *CARDIS 2014*, volume 8968 of *LNCS*, pages 233–244. Springer, 2014.
14. C. Dobraunig, F. Koeune, S. Mangard, F. Mendel, and F.-X. Standaert. Towards fresh and hybrid re-keying schemes with beyond birthday security. In *CARDIS 2015*, volume 9514 of *LNCS*, pages 225–241. Springer, 2015.
15. Y. Dodis and K. Pietrzak. Leakage-resilient pseudorandom functions and side-channel attacks on feistel networks. In *CRYPTO 2010*, volume 6223 of *LNCS*, pages 21–40. Springer, 2010.
16. S. Dziembowski and K. Pietrzak. Leakage-resilient cryptography. In *FOCS 2008*, pages 293–302. IEEE Computer Society, 2008.
17. S. Faust, K. Pietrzak, and J. Schipper. Practical leakage-resilient symmetric cryptography. In *CHES 2012*, volume 7428 of *LNCS*, pages 213–232. Springer, 2012.
18. O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.
19. Y. Ishai, A. Sahai, and D. Wagner. Private circuits: Securing hardware against probing attacks. In D. Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 463–481. Springer, 2003.
20. P. C. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In *CRYPTO '99*, volume 1666 of *LNCS*, pages 388–397. Springer, 1999.
21. S. Mangard, E. Oswald, and T. Popp. *Power analysis attacks – Revealing the secrets of smart cards*. Springer, 2007.
22. M. Medwed, C. Petit, F. Regazzoni, M. Renauld, and F.-X. Standaert. Fresh re-keying II: securing multiple parties against side-channel and fault attacks. In *CARDIS 2011*, volume 7079 of *LNCS*, pages 115–132. Springer, 2011.
23. M. Medwed, F.-X. Standaert, J. Großschädl, and F. Regazzoni. Fresh re-keying: Security against side-channel and fault attacks for low-cost devices. In *AFRICACRYPT 2010*, volume 6055 of *LNCS*, pages 279–296. Springer, 2010.

24. M. Medwed, F.-X. Standaert, and A. Joux. Towards super-exponential side-channel security with efficient leakage-resilient prfs. In *CHES 2012*, volume 7428 of *LNCS*, pages 193–212. Springer, 2012.
25. F. Mendel, B. Mennink, V. Rijmen, and E. Tischhauser. A simple key-recovery attack on McOE-X. In *CANS 2012*, volume 7712 of *LNCS*, pages 23–31. Springer, 2012.
26. A. Moradi, A. Barenghi, T. Kasper, and C. Paar. On the vulnerability of FPGA bitstream encryption against power analysis attacks: extracting keys from Xilinx Virtex-II FPGAs. In *ACM CCS 2011*, pages 111–124, 2011.
27. A. Moradi and T. Schneider. Improved side-channel analysis attacks on xilinx bitstream encryption of 5, 6, and 7 series. In F. Standaert and E. Oswald, editors, *COSADE 2016*, volume 9689 of *LNCS*, pages 71–87. Springer, 2016.
28. C. O'Flynn and Z. D. Chen. Chipwhisperer: An open-source platform for hardware embedded security research. In *COSADE 2014*, pages 243–260, 2014.
29. O. Pereira, F.-X. Standaert, and S. Vivek. Leakage-resilient authentication and encryption from symmetric cryptographic primitives. In *CCS 2015*, pages 96–108. ACM, 2015.
30. P. Pessl and S. Mangard. Enhancing side-channel analysis of binary-field multiplication with bit reliability. In *CT-RSA 2016*, volume 9610 of *LNCS*, pages 255–270. Springer, 2016.
31. K. Pietrzak. A leakage-resilient mode of operation. In *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 462–482. Springer, 2009.
32. B. Preneel and P. C. van Oorschot. On the security of two MAC algorithms. In *EUROCRYPT '96*, volume 1070 of *LNCS*, pages 19–32. Springer, 1996.
33. E. Prouff and M. Rivain. Masking against side-channel attacks: A formal security proof. In *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 142–159. Springer, 2013.
34. J.-J. Quisquater and D. Samyde. Electromagnetic analysis (EMA): measures and counter-measures for smart cards. In *E-smart 2001*, volume 2140 of *LNCS*, pages 200–210. Springer, 2001.
35. F.-X. Standaert, O. Pereira, Y. Yu, J.-J. Quisquater, M. Yung, and E. Oswald. Leakage resilient cryptography in practice. In *Towards Hardware-Intrinsic Security – Foundations and Practice*, Information Security and Cryptography, pages 99–134. Springer, 2010.
36. The CAESAR committee. CAESAR: Competition for authenticated encryption: Security, applicability, and robustness. http://competitions.cr.yp.to/, 2014.
37. Y. Yu and F.-X. Standaert. Practical leakage-resilient pseudorandom objects with minimum public randomness. In *CT-RSA 2013*, volume 7779 of *LNCS*, pages 223–238. Springer, 2013.

A Keccak[400] Instance

The authenticated encryption scheme and its permutation-based instance Isap were designed to be secure against passive side-channel attacks. For practical use, we will now give instance parameters for the four algorithms IsapEnc, IsapMac, IsapRk1 and IsapRk2. Based on the specified parameters we will then give implementation results. Our results show that a hardware implementation of Isap yields performance and area figures comparable to state-of-the-art

schemes, while extending resistance against DPA to settings allowing multiple decryption.

The parameters of Isap have been chosen such that permutations are allowed to leak up to 16 bits per invocation. However, the concrete choice of the parameters for instances of our permutation-based designs also depends on the desired security against cryptographic attacks. Compliant with the CAESAR call, we state as our security level the intended number of bits of security. The *cryptographic* security levels of the encryption scheme IsapEnc, message authentication code IsapMac, and the two re-keying functions IsapRk1, IsapRk2, are summarized for a b-bit permutation p in Table 2.

Table 2: Security level (in bits) for the different building blocks of Isap.

Function	Security (bits)	References
IsapEnc	$\min(k, b/2, c)$	[1]
IsapMac	$\min(k, c/2, n)$	[4–6]
IsapRk1	$\min(k, b/2, n)$	
IsapRk2	$\min(k, b/2, c, n)$	[1]

A.1 Parameter Selection

Based on Table 2, we propose using the permutation Keccak[400] to target a 128-bit security level. Both its permutation size $b = 400$ and a chosen capacity of $c = 256$ make Keccak[400] a suitable choice to meet the security requirements for IsapEnc and IsapMac. Moreover, Keccak[400] also provides adequate security for the two re-keying functions IsapRk1 and IsapRk2. Our choice for the number of rounds for the permutation used in our primitives relies on the CEASAR candidate Keyak [9], which uses 12 rounds, and on Keccak[400] using 20 rounds [7]. Since the security of IsapMac is based on the security of the underlying hash function, we decided to rely on the scrutinized, original Keccak [7] proposal and use a 20-round version of the Keccak[400] permutation. In contrast, IsapEnc and IsapRk2 inject the secret key before the first permutation call, significantly restricting the possible attack vectors. Thus, we followed the proposal of Keyak [9] (although it does only propose variants using the larger 800 and 1600-bit permutations) and use 12 rounds. The only non-sponge design IsapRk1 uses a very conservative number of 20 rounds. Since we cannot rely here on third-party analysis as in the case of our constructions that are similar to Keyak [9], we want to use a very strong permutation.

Table 3 summarizes the recommended capacity for the different building blocks along with the required number of rounds for the permutation. For domain separation of the four algorithms, we suggest using distinct IVs and/or padding.

The full Keccak[400]-based Isap instance also maintains 128-bit security for the permutation leaking up to 16 bits per invocation. This is based on each

Table 3: Parameters for ISAP with 128-bit security using the Keccak[400] permutation ($k = 128$, $b = 400$).

Function	Capacity c	max. State Leakage	Rounds	
IsapEnc	256	128	12	[9]
IsapMac	256	–	20	[7]
IsapRk1	–	72	20	[7]
IsapRk2	399	271	12	[9]

algorithm's maximum tolerated state leakage given in Table 3, which can be determined using the approach discussed in Section 3.3 and the security levels in Table 2. In case of IsapEnc, the choice of Keccak[400] with capacity $c = 256$ would allow the implementation to leak about 128 bits of its secret state as long as the session key is not concerned. This suggests a maximum tolerable leakage of 64 bits per invocation of the permutation. However, also leakage of the session key has to be considered in the initial permutation of IsapEnc, i.e., the permutation processing session key k_1, nonce n, and iv. Allowing the initial permutation to leak 64 bits could—in the worst case—reduce the security of the session key k_1 by 64 bits. Yet, to cope with such leakage, we suggest deriving 144-bit session keys from 128-bit master keys. This allows the permutation directly using the session key to leak 16 bits thereof without falling below 128-bit security. For IsapMac, we do not care about any leakage in the hashing part since all values are known to the attacker anyway. However, similar to IsapEnc, 144-bit session keys are recommended in IsapMac to allow 16-bit session key leakage in the last permutation.

Similarly, the two re-keying functions IsapRk1 and IsapRk2 allow for a certain 2λ-bit leakage of their internal state. For IsapRk1, 72 bits of the internal state can be leaked (36 bits per permutation invocation). For IsapRk2 with an r-bit rate, security is determined by $\min(k, (b - \min(r, 2\lambda))/2, c - 2\lambda, n)$. Therefore, for the rate $r = 1$, up to 271 bits of the internal state can be leaked. However, IsapRk2 is a 2-limiting design and not inherently secure against DPA. Therefore, allowing such larger amounts of leakage will be necessary in practice. To further increase security, we recommend to store the 400-bit expanded key that is initially used in either of the two re-keying functions instead of the 128-bit master-key. Using the 400-bit expanded key avoids leakage of master-key bits in the initial permutation and increases the overall security due to the larger secret state that is used within the re-keying functions. Also note that load-time leakage of the master key can be considered as state leakage before the initial permutation.

A.2 Implementation Results and Comparison

We first discuss the implementation of the re-keying functions followed by the full authenticated encryption scheme.

Re-keying Function. The re-keying functions IsapRk1 and IsapRk2 were implemented according to the instance parameters in Section A.1. Both designs compute one permutation round per cycle and were synthesized in an UMC 130 nm technology. For comparison, we also implemented and synthesized a GGM-based re-keying function [35] based on an AES capable of one round per cycle. The respective results shown in Table 4 are supplemented with synthesis results stated in the literature for re-keying using a masked polynomial multiplication over a finite field [22, 23].

Table 4: Implementation results for secure re-keying functions (130 nm).

Function	Area [kGE]	Frequency [MHz]	Runtime[1] [cycles]	[μs]
IsapRk1	8.5	172	2 709	15.8
IsapRk2	7.7	212	1 677	7.9
AES-GGM [35]	11.2	101	1 536	15.2
PolyMult [23]	10.2	–	1 160	–

[1)] Runtime is given for 128-bit nonces.

The results in Table 4 suggest that both IsapRk1 and IsapRk2 give a smaller re-keying function than using an AES-based GGM tree. While IsapRk2 is twice as fast as AES-GGM, IsapRk1 is similarly fast. Note however that IsapRk1 provides inherent security to DPA attacks while AES-GGM and IsapRk2 are 2-limiting. Moreover, both IsapRk1 and IsapRk2 provide a significant security margin to cope with SPA leakage.

Compared to the polynomial multiplication with 3rd order masking in [23], the re-keying functions IsapRk1 and IsapRk2 have a smaller area footprint and do not require dedicated DPA countermeasures, but were designed in view of preventing DPA scenarios at all. Besides, the weak algebraic structure of the multiplication opens the door to combined attacks on the re-keying function and the encryption [30] as well as to TMTO attacks [10, 13, 25] that recover the master key.

Apart from the re-keying functions listed in Table 4, there are also re-keying functions based on leakage-resilient pseudo-random functions [2, 24] (LR-PRFs). We ommitted such re-keying functions based on LR-PRFs as their security can hardly be compared to the ones of IsapRk1 and IsapRk2 as it is mainly based on the assumption that the attacker is not able to distinguish the leakage of different hardware components on a chip. However, the 7.3 kGE implementation of LR-PRFs performs re-keying in 0.5 μs at 338 MHz and is thus faster than both Isap re-keying functions. Yet, concerning applications like FPGA bitfile encryption, re-keying will typically not be the bottleneck.

Authenticated Encryption Scheme. The authenticated encryption scheme consisting of IsapEnc and IsapMac was implemented in two variants, where the first variant uses IsapRk1 and the second uses IsapRk2. Both variants employ a single Keccak[400] permutation that performs one round per cycle. The synthesis results using a 130 nm UMC technology are shown in Table 5. The runtime is given for encryption and authentication of a single 128-bit data block. Hereby, secure initialization amounts to $36.35\mu s$ and $22.35\mu s$ for IsapAe-Rk1 and IsapAe-Rk2, respectively. However, this initialization part becomes negligible for bulk data processing. For a single 128-bit data block, it shows that IsapAe-Rk2 is 11% smaller and 38% faster than IsapAe-Rk1. The main reason for this performance penalty lies in the higher number of permutation rounds for initialization with IsapRk1.

Table 5: Implementation of the AE modes (130 nm).

Function	Area [kGE]	Frequency [MHz]	Runtime[1] [cycles]	[μs]
IsapAe-Rk1	15.8	169	6 171	36.5[2]
IsapAe-Rk2	14.0	171	3 853	22.5[3]

[1)] Runtime is given for 144-bit nonces.
[2)] Secure initialization: $36.35\mu s$.
[3)] Secure initialization: $22.35\mu s$.

While IsapAe is designed to be secure against DPA also for multiple decryptions, there are no other schemes fulfilling this security property. On the other hand, leakage-resilient schemes such as in [29, 35] come with a formal security proof we do not provide. It is therefore hard to compare our results fairly with related work.

However, we yet implemented a hardware module comprising the leakage-resilient encryption scheme in [35] and the leakage-resilient MAC in [29] to provide authenticated encryption. The module uses the GGM-based re-keying function that shares with both the MAC and the encryption scheme a single AES module that computes one round per cycle. The module is sized 13.1 kGE and processes one 128-bit data block in 29.9 μs. Compared to the two variants of IsapAe, the AES-based leakage-resilient encryption and MAC [29, 35] thus gives quite similar results both in terms of area and runtime, even though IsapAe uses a significantly larger state.